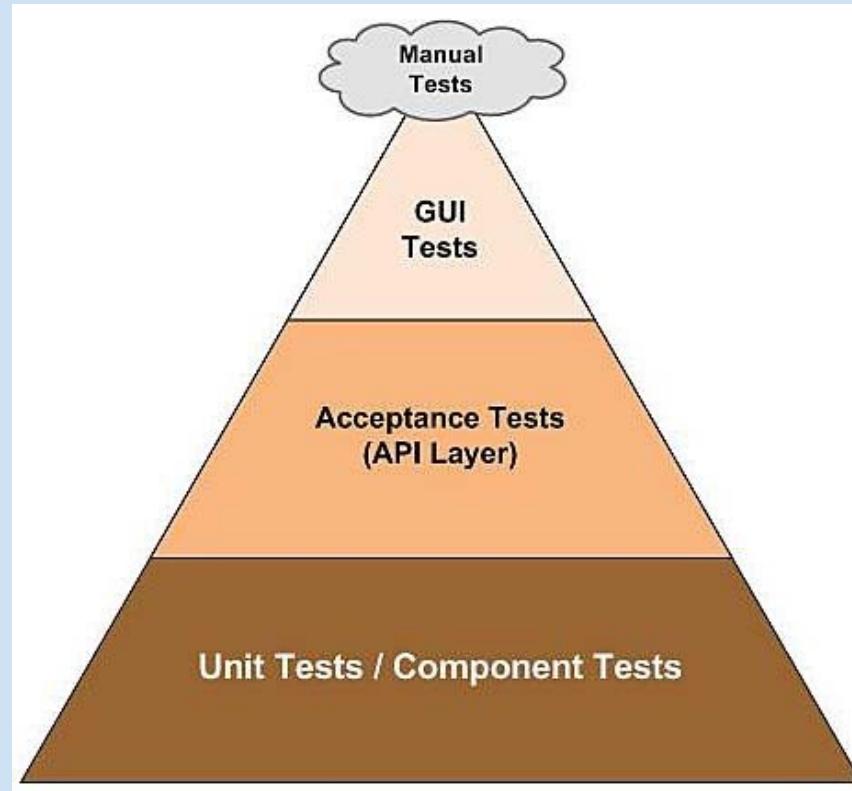


Agile Swift

Godfrey Nolan

RIIS LLC

Agenda

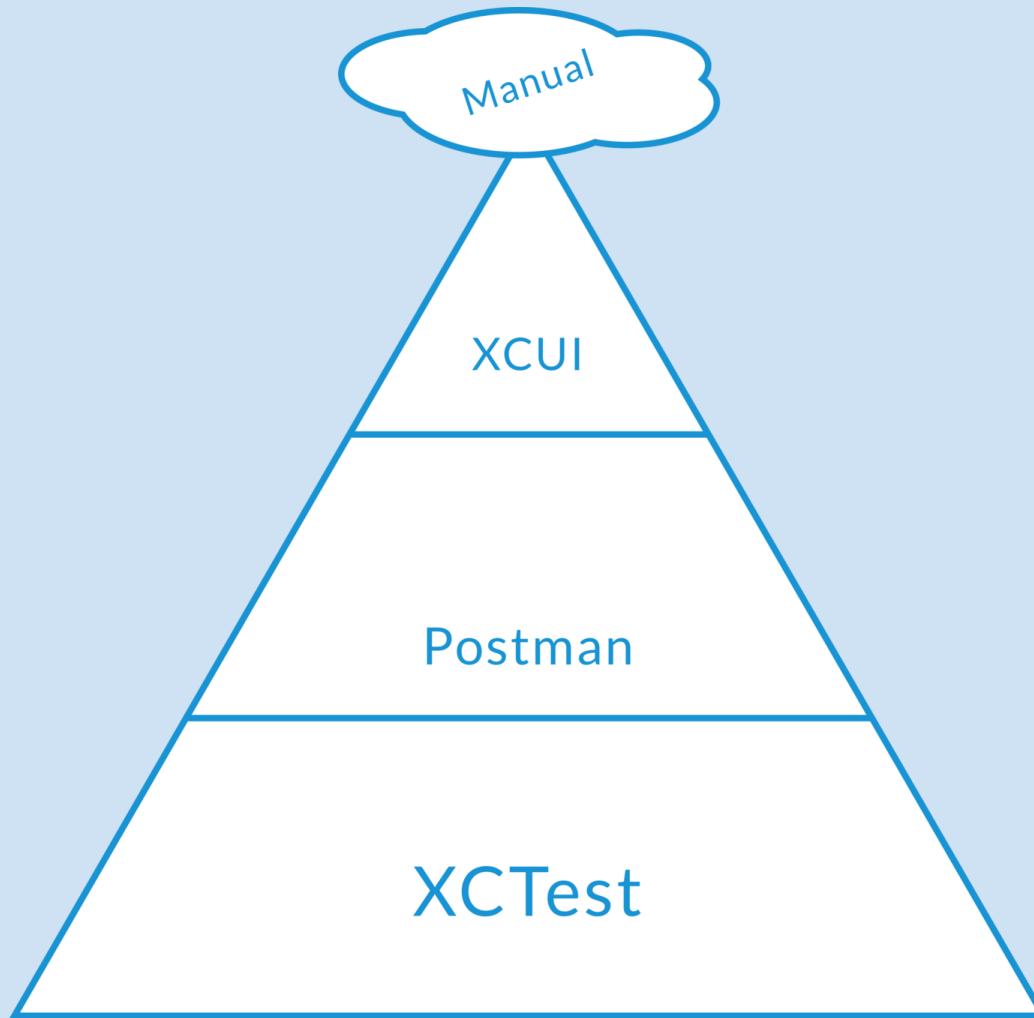




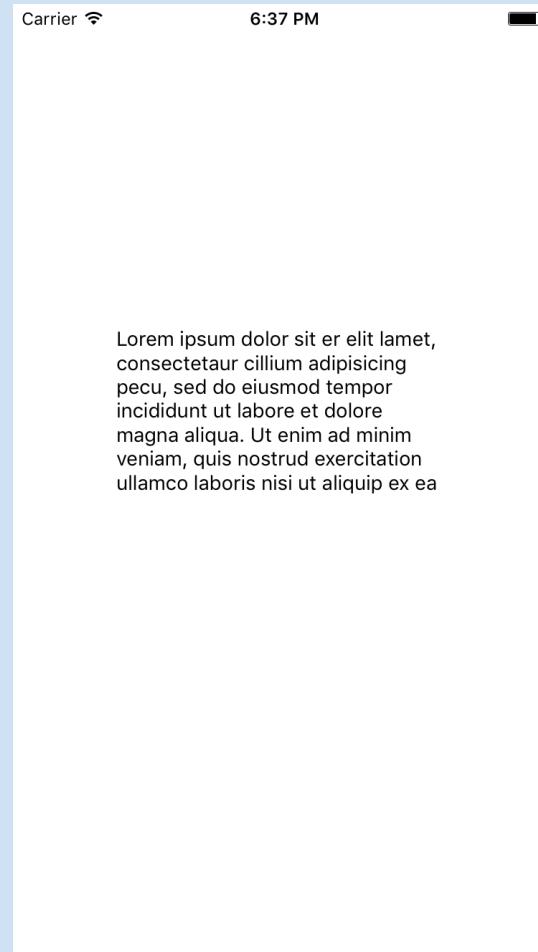
Agile Testing Benefits

- Catch more mistakes
- Confidently make more changes
- Built in regression testing
- Extend the life of your codebase
- Better predictability & reliability

Agile Swift



Hello World



Lab 1 - HelloWorld

1. Open Xcode
2. Make sure Xcode is 8.x
3. Open Main.storyboard
4. Search for TextView in the Object Library
5. Drag over to Storyboard
6. Run

Swift Unit Test - XCTest

Choose options for your new project:

Product Name: Calculator

Organization Name: riis

Organization Identifier: com.riis

Bundle Identifier: com.riis.Calculator

Language: Swift

Devices: iPhone

Use Core Data

Include Unit Tests

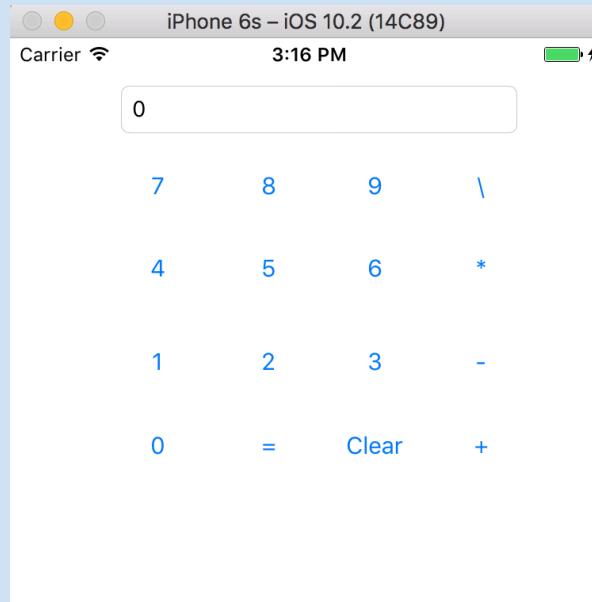
Include UI Tests

Cancel

Previous

Next

Swift Unit Test - XCTest



The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure under the "Calculator" target:
 - Calculator folder contains: AppDelegate.swift, ViewController.swift, Main.storyboard, Assets.xcassets, LaunchScreen.storyboard, Info.plist, and CalculatorModel.swift (selected).
 - CalculatorTests folder contains: CalculatorTests.swift and Info.plist.
 - CalculatorUITests folder.
 - Products folder.
- Editor:** Displays the content of the selected file, `CalculatorModel.swift`. The code defines a `CalculatorModel` class with methods for addition, subtraction, multiplication, and division.
- Toolbar:** Standard Xcode toolbar with icons for file operations.
- Status Bar:** Shows "Calculator | Build Calculator: Succeeded | 12/4/16 at 12:55 PM".
- Notification:** A yellow warning icon with the number "1" in the top right corner.

```
1 ///
2 //  CalculatorModel.swift
3 //  Calculator
4 //-
5 //  Created by User on 12/4/16.
6 //  Copyright © 2016 example. All rights reserved.
7 //
8
9 import Foundation
10
11
12 class CalculatorModel {
13     var a: Int!
14     var b: Int!
15
16     func add(_ a:Int, _ b:Int) -> Int {
17         return a + b
18     }
19
20     func sub(_ a:Int, _ b:Int) -> Int {
21         return a - b
22     }
23
24     func mul(_ a:Int, _ b:Int) -> Int {
25         return a * b
26     }
27
28     func div(_ a:Int, _ b:Int) -> Int {
29         guard b != 0 else {
30             return 0
31         }
32         return a / b
33     }
34 }
35 }
```

Calculator > iPhone 6s

Running Calculator on iPhone 6s

11

Calculator Scene

View Controller

Top Layout Guide

Bottom Layout Guide

View

F Results Fld

B 7

B 8

B 9

B \

B 4

B 5

B 6

B *

B 1

B 2

B 3

B -

B 0

B =

B Clear

B +

First Responder

Exit

Storyboard Entry Point

View as: iPhone 7 (wC hR) 100% +

Quick Help

Declaration @interface UIView : UIResponder <NSCoding, UIAppearance, UIAppearanceContainer, UIDynamicItem, UITraitEnvironment, UICoordinateSpace, UIFocusItem, CALayerDelegate>

Description The UIView class defines a rectangular area on the screen and the interfaces for managing the content in that area.

At runtime, a view object handles the rendering of any content in its area and also handles any interactions with that content. The UIView class itself provides basic behavior for filling its rectangular area with a background color. More sophisticated content can be presented by subclassing UIView and implementing the necessary drawing and event-handling code yourself. The UIKit framework also includes a set of standard subclasses that range from simple buttons to complex tables and can be used as-is. For example, a UILabel object

View Controller - A controller that manages a view.

Storyboard Reference - Provides a placeholder for a view controller in an external storyboard.

Navigation Controller - A controller that manages navigation through a hierarchy of views.

Filter

Calculator

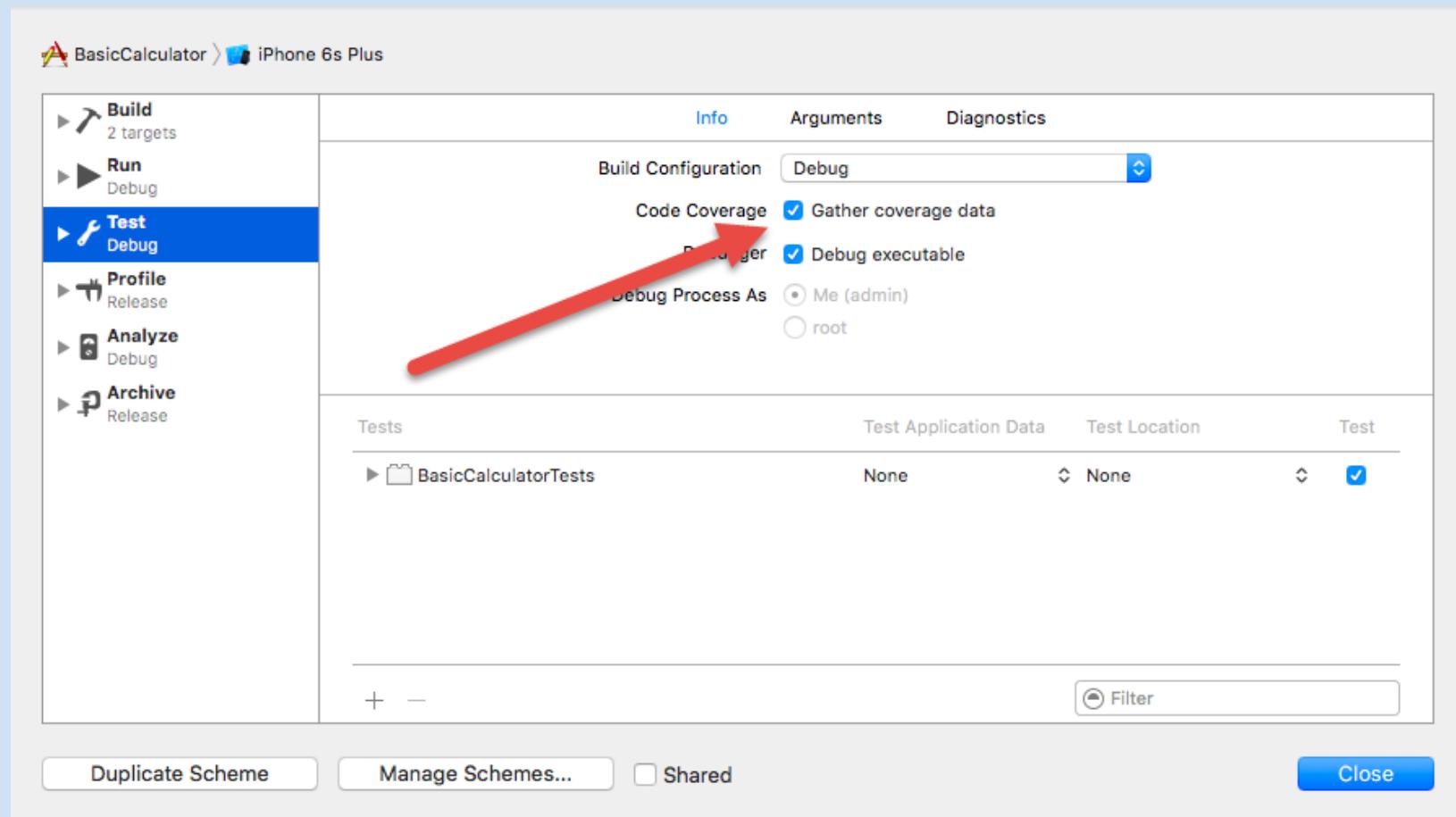
Lab 2 - Calculator

0. Download code from <https://github.com/godfreynolan/CodeCraftsman>
1. Open Calculator in iOS Folder
2. Click on Calculator Tests
3. Open CalculatorTests.swift
4. Run tests
5. Open Test Navigator
6. Run tests again
7. Open Report Navigator
8. View Logs->All Messages
9. Add more tests to CalculatorTests for sub, mul, div functions

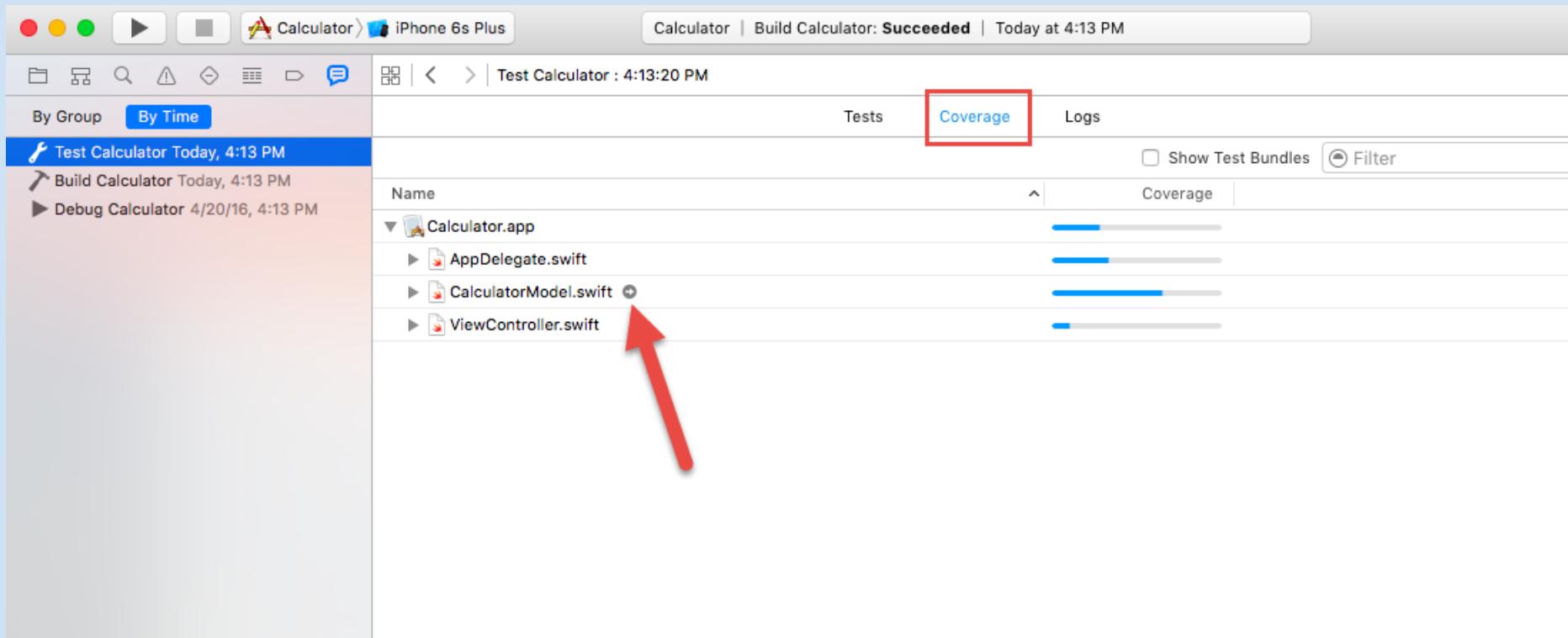
Unit Testing Assertions

Assertion	Description
XCTAssert	Tests that two values are the same
XCTAssertEqual	Tests that two values are equal
XCTAssertEqualWithAccuracy	Tests that two floating point values (a,b) are equal within a tolerance of c
XCTAssertFalse	Tests if a Boolean condition is false
XCTAssertTrue	Tests if a Boolean condition is true
XCTAssertGreaterThan	Tests that one value is greater than the other
XCTAssertGreaterThanOrEqualTo	Tests that one value is greater or equal to the other
XCTAssertLessThan	Tests that one value is less than the other
XCTAssertLessThanOrEqualTo	Tests that one value is less than or equal to the other
XCTAssertNil	Tests that an object is nil
XCTAssertNotEqual	Tests that two values are not equal
XCTAssertNotEqualWithAccuracy	Tests that two floating point values (a,b) are not equal within a tolerance of c
XCTAssertNotNil	Tests that an object is not nil

Unit Testing Code Coverage



Unit Testing Code Coverage



The screenshot shows the Xcode IDE interface. At the top, there's a toolbar with standard Mac OS X window controls (red, yellow, green, close, minimize, zoom) and a tab bar showing "Calculator" and "iPhone 6s Plus". To the right of the tab bar is a status bar displaying "Calculator | Build Calculator: Succeeded | Today at 4:13 PM".

The main area is a code editor with a sidebar on the left containing a list of recent tasks:

- Test Calculator Today, 4:13 PM (selected)
- Build Calculator Today, 4:13 PM
- Debug Calculator 4/20/16, 4:13 PM

The code editor displays the file "CalculatorModel.swift" under the "Calculator" group. The code defines a class "CalculatorModel" with methods for addition, subtraction, multiplication, division, and a constructor. The "add(a:Int, b:Int) -> Int" method is currently selected.

```
// CalculatorModel.swift
// Calculator
//
// Created by Admin on 4/16/16.
// Copyright © 2016 riis. All rights reserved.

import Foundation

class CalculatorModel {

    var a: Int
    var b: Int

    init(a:Int, b:Int){
        self.a = a
        self.b = b
    }

    func add(a:Int, b:Int) -> Int {
        return a + b
    }

    func sub(a:Int, b:Int) -> Int {
        return a - b
    }

    func mul(a:Int, b:Int) -> Int {
        return a * b
    }

    func div(a:Int, b:Int) -> Int {
        guard b != 0 else {
            return 0
        }
        return a / b
    }
}
```

Code Coverage Lab

1. Open Calculator in iOS Folder
2. Enable Code Coverage
3. Run tests
4. View Code coverage results

Unit Testing Code Coverage



SLATHER
Apply tests liberally

Coverage for "CalculatorModel.swift" : 65.00%
(13 of 20 relevant lines covered)

Calculator/Calculator/CalculatorModel.swift

```
1 // CalculatorModel.swift
2 // Calculator
3 //
4 // Created by Admin on 4/16/16.
5 // Copyright © 2016 riis. All rights reserved.
6 //
7 //
8 import Foundation
9
10 class CalculatorModel {
11
12     var operandOne: Int
13     var operandTwo: Int
14
15     init(operandOne: Int, operandTwo: Int) {
16         self.operandOne = operandOne
17         self.operandTwo = operandTwo
18     }
19
20
21     func add(operandOne: Int, operandTwo: Int) -> Int {
22         return operandOne + operandTwo
23     }
24
25     func sub(operandOne: Int, operandTwo: Int) -> Int {
26         return operandOne - operandTwo
27     }
28
29     func mul(operandOne: Int, operandTwo: Int) -> Int {
30         return operandOne * operandTwo
31     }
32
33     func div(operandOne: Int, operandTwo: Int) -> Int {
34
35         guard operandTwo != 0 else {
36             return 0
37         }
38         return operandOne / operandTwo
39     }
40 }
```

\$ brew install ruby

\$ gem install slather

\$ slather coverage --html --scheme XcodeSchemeName path/to/project.xcodeproj

\$ slather coverage --html --scheme Calculator Calculator/Calculator.xcodeproj

Code Coverage Lab #2

1. Install slather

```
brew install ruby
gem install slather
```
2. From the dir above Calculator run

```
slather coverage --html --scheme Calculator Calculator.xcodeproj
```
3. Find the html output

Swift on other platforms

```
[RIIS-MBPRO-13:HelloWorld2 User$ swift package init
Creating library package: HelloWorld2
Creating Package.swift
Creating .gitignore
Creating Sources/
Creating Sources/HelloWorld2.swift
Creating Tests/
Creating Tests/LinuxMain.swift
Creating Tests>HelloWorld2Tests/
Creating Tests>HelloWorld2Tests>HelloWorld2Tests.swift
RIIS-MBPR0-13:HelloWorld2 User$ ]
```

Swift on other platforms

```
RIIS-MBPRO-13:HelloWorld2 User$ swift test
Compile Swift Module 'HelloWorld2' (1 sources)
Compile Swift Module 'HelloWorld2Tests' (1 sources)
Linking ./build/debug/HelloWorld2PackageTests.xctest/Contents/MacOS>HelloWorld2PackageTests
Test Suite 'All tests' started at 2017-04-07 18:45:50.743
Test Suite 'HelloWorld2PackageTests.xctest' started at 2017-04-07 18:45:50.744
Test Suite 'HelloWorld2Tests' started at 2017-04-07 18:45:50.744
Test Case '-[HelloWorld2Tests.HelloWorld2Tests testExample]' started.
Test Case '-[HelloWorld2Tests.HelloWorld2Tests testExample]' passed (0.002 seconds).
Test Suite 'HelloWorld2Tests' passed at 2017-04-07 18:45:50.747.
    Executed 1 test, with 0 failures (0 unexpected) in 0.002 (0.002) seconds
Test Suite 'HelloWorld2PackageTests.xctest' passed at 2017-04-07 18:45:50.747.
    Executed 1 test, with 0 failures (0 unexpected) in 0.002 (0.003) seconds
Test Suite 'All tests' passed at 2017-04-07 18:45:50.747.
    Executed 1 test, with 0 failures (0 unexpected) in 0.002 (0.004) seconds
```

API Testing - Postman

The screenshot shows the Postman interface for API testing. At the top, there's a header with 'GET' (dropdown), 'http://ec2-204-236-211-33.compute-1.amazonaws.com:8080/companies/1/routes/1/Northbound/weekday/1/stops' (URL input), 'Params' (button), 'Send' (blue button), and 'Save' (button). Below the URL input is an 'Authorization' dropdown. To the right of the 'Send' button are 'Cookies' and 'Code' buttons.

The main area contains a code editor with the following content:

```
1 tests["Body has stopID"] = responseBody.has("stopID");
2 tests["Body has stopName"] = responseBody.has("stopName");
3 tests["Body has latitude"] = responseBody.has("latitude");
4 tests["Body has longitude"] = responseBody.has("longitude");
5
6 var jsonData = JSON.parse(responseBody);
7
8 tests["The first stopID equals String 22362"] = jsonData[0]
    .stopID === "22362";
9 tests["The first stopName equals String METRO AIRPORT
    McNamara TERMINAL"] = jsonData[0].stopName === "METRO
    AIRPORT McNamara TERMINAL";
10 tests["The first order equals String 1"] = jsonData[0].order
    === 1;
```

To the right of the code editor is a sidebar titled 'SNIPPETS' with the following items:

- Clear a global variable
- Clear an environment variable
- Response body: Contains string
- Response body: Convert XML body to a JSON Object
- Response body: Is equal to a string
- Response body: JSON value check
- Response headers: Content-Type header check
- Response time is less than 200ms
- Set a global variable

API Testing - Newman

```
RIIS-MBPR0-13:Android User$ newman run ETAJson.postman_collection.json
newman
```

```
ETAJson
```

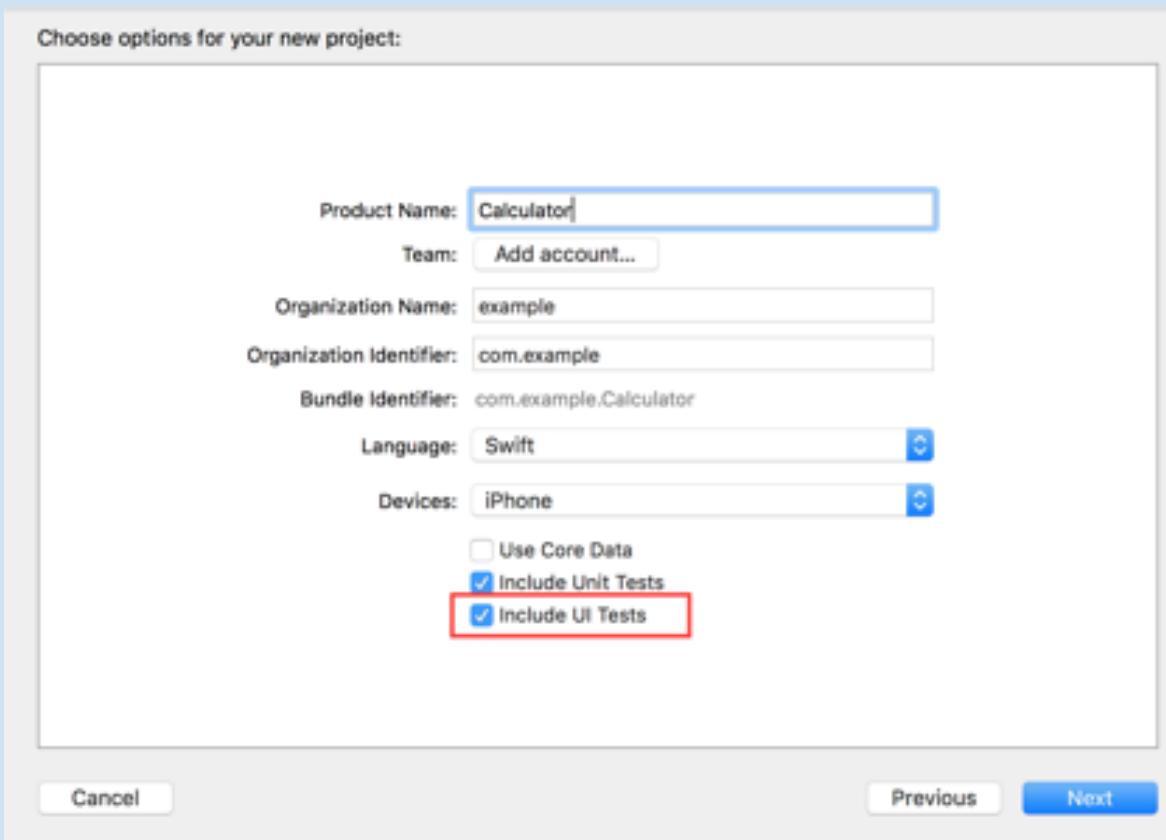
```
→ http://ec2-204-236-211-33.compute-1.amazonaws.com:8080/companies/1/routes/1/Northbound/weekday/1/stops
  GET http://ec2-204-236-211-33.compute-1.amazonaws.com:8080/companies/1/routes/1/Northbound/weekday/1/stops [200 OK, 1.74KB,
103ms]
✓ Body has stopID
✓ Body has stopName
✓ Body has latitude
✓ Body has longitude
✓ The first stopID equals String 2236
✓ The first stopName equals String METRO AIRPORT MCNAMARA TERMINAL
✓ The first order equals String 1
```

	executed	failed
iterations	1	0
requests	1	0
test-scripts	1	0
prerequest-scripts	0	0
assertions	7	0
total run duration:	152ms	
total data received:	1.59KB (approx)	
average response time:	103ms	

API Testing - Newman

1. Download nodejs from <https://nodejs.org/en/download/package-manager>
2. Install newman
`npm install newman --global;`
3. Run the tests
`newman run ETAJson.postman_collection.json`

Swift GUI Testing - XCUI



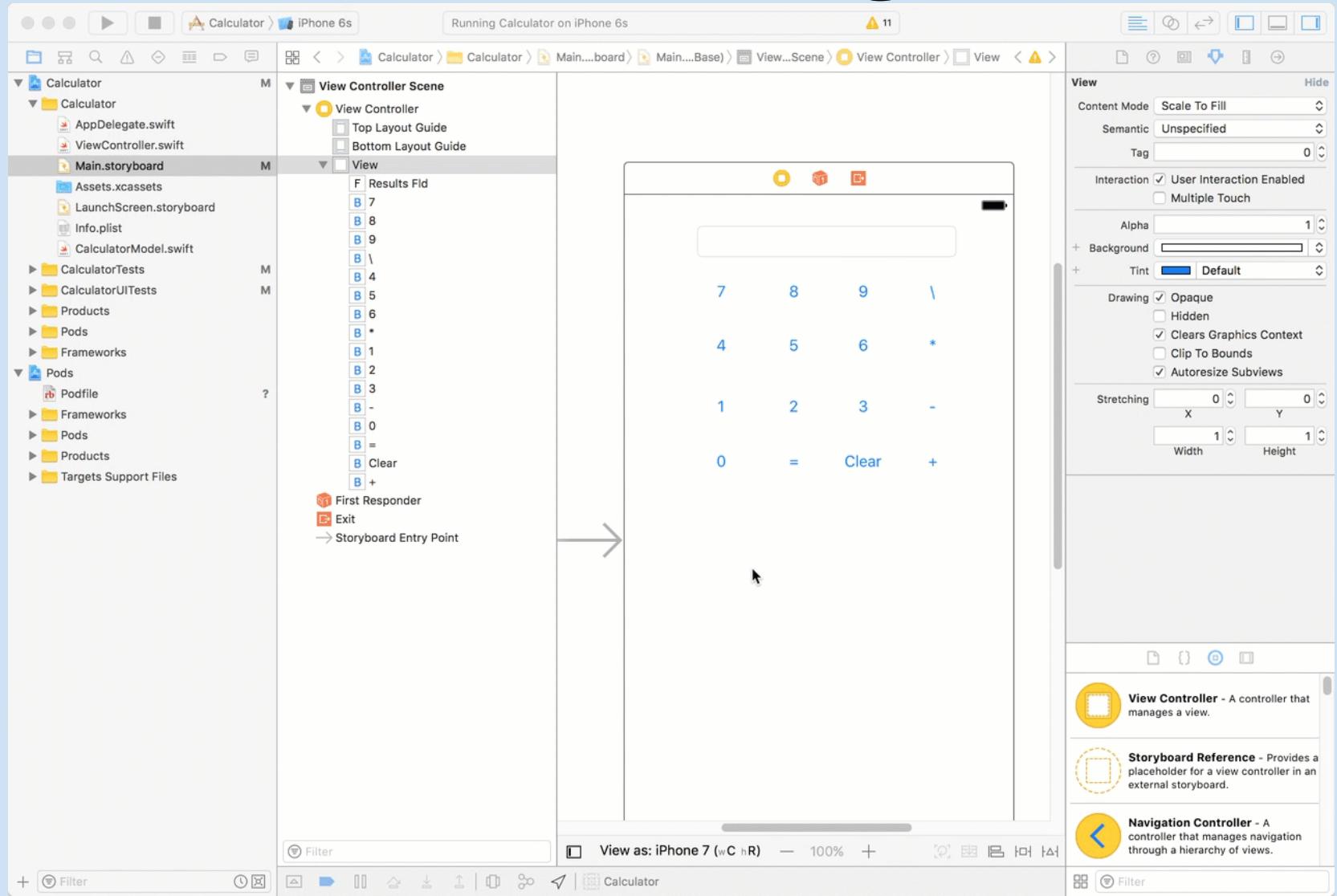
Swift GUI Testing - XCUI

- User Interface Testing
- Two Options
 - Recorded Tests
 - Roll Your Own

Swift GUI Testing - XCUI

- XCUIApplication
 - Launch the app
- XCUIElementQuery
 - Find the XCUI Element to test
- XCUIElement
 - Perform Test

Swift GUI Testing - XCUI



Calculator > iPhone 6s Calculator | Clean Calculator: Succeeded | Today at 3:49 PM

Calculator M CalculatorUITests M CalculatorUITests

```
4 //  
5 // Created by User on 12/4/16.  
6 // Copyright © 2016 example. All rights reserved.  
7 //  
8  
9 import XCTest  
10  
◇ 11 class CalculatorUITests: XCTestCase {  
12  
    override func setUp() {  
        super.setUp()  
        continueAfterFailure = false  
        XCUIApplication().launch()  
    }  
18  
    override func tearDown() {  
        super.tearDown()  
    }  
22  
    ◇ 23 func testExample() {  
        // click here to record  
  
        let app = XCUIApplication()  
        app.buttons["Clear"].tap()  
        app.buttons["7"].tap()  
        app.buttons["*"].tap()  
        app.buttons["2"].tap()  
        app.buttons["="].tap()  
  
        let textField = app.otherElements.children  
            (matching: . textField).element  
        XCTAssertEqual(textField.value as! String, "14")  
36  
37    }  
38  
39 }
```

Carrier iPhone 6s – iOS 10.2 (14C89) 3:50 PM

Watch Extras iCloud Drive Calculator Calculator...

Safari Messages

No Matches

Swift GUI Testing Lab

1. Open Calculator project in xcode
2. Go to GUI Tests
3. Create new test file
4. record tests
5. Add assertion
6. Replay tests

Putting It All Together

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Project name HelloWorld

Description

[Plain text] [Preview](#)

Discard old builds [?](#)

GitHub project

Project url <https://github.com/gnolanltu>HelloWorld/> [?](#)

[Advanced...](#)

This project is parameterized [?](#)

Throttle builds [?](#)

Disable this project [?](#)

Execute concurrent builds if necessary [?](#)

[Advanced...](#)

Putting It All Together

① www.cimgf.com/2015/05/26/setting-up-jenkins-ci-on-a-mac-2/

 COCOA IS MY GIRLFRIEND
CIMGF.COM

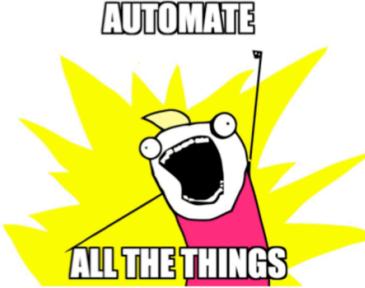
Search...

About
Cocoa Code Snippets
ZDS Code Style Guide

Categories
Advanced
Announcement
App Distribution
Apple
AppStore
Audio
Beginner
Blogging
Cocoa
Cocoa Touch
Coding Practice
Community
Continuous Integration
Core Animation
Core Data
Core Video
Development Environment
Frameworks

SETTING UP JENKINS CI ON A MAC
by Eric Cerney

26 MAY 2015
Continuous Integration


AUTOMATE
ALL THE THINGS

Setting up a CI server can be nothing but pain and suffering, yet many of us take on this challenge to save time in the long run. Some things I would rather do than set up a CI server:

1. Get a root canal at my desk waiting for Xcode SourceKit crashes to calm down.
2. Submit an Apple Radar for one of my many Xcode crashes.
3. Write a tutorial on how to set up a Jenkins CI server. (Oh hey!)

Whether you're being a good developer and doing your unit testing, are tired of co-workers breaking your code, or have clients that can't live without the latest and greatest build, CI will be your new best friend and butler.

There are tons of posts floating around the internet with bits and pieces on this topic; but none were comprehensive enough for me to get a server up and running. Because of this, I decided to write a thorough guide that will walk you through setting up a [Jenkins CI Server](#) on your machine of choice. In this instance we used a Mac Mini.

Note that our end goal with Jenkins was to build, test, and distribute apps for both the iOS and Android platforms. Some of the setup in this guide is aimed at accomplishing this goal.



Putting It All Together

1. Download the Jenkins Mac OS X native package from <http://jenkins-ci.org>.
2. Double click the .pkg file to install Jenkins.
3. Once done, your browser will open to <http://localhost:8080> where Jenkins lives.
4. Make the Jenkins user an admin:

```
sudo dseditgroup -o edit -a jenkins -t user admin
```
5. Add the Jenkins user to the developer group:

```
sudo dscl . append /Groups/_developer GroupMembership jenkins
```
6. Make the Jenkins user automatically login when the computer is restarted
7. unload Jenkins as a Daemon:

```
sudo launchctl unload /Library/LaunchDaemons/org.jenkins-ci.plist
```
8. move the .plist file, which defines how Jenkins will run, to the LaunchAgents folder:

```
sudo mv /Library/LaunchDaemons/org.jenkins-ci.plist /Library/LaunchAgents/
```
9. Edit the plist file:

```
sudo vim /Library/LaunchAgents/org.jenkins-ci.plist
/* Remove the following lines */
<key>SessionCreate</key
<true />
```
10. reload the Launch Agent to restart Jenkins:

```
sudo launchctl load /Library/LaunchAgents/org.jenkins-ci.plist
```

FIRST Principles

- F(ast)
- I(solated)
- R(epetatable)
- S(elf-verifying)
- T(imely) i.e. TDD not TAD

FIRST Principles - Fast

Feature	Small	Medium	Large
Network access	No	localhost only	Yes
Database	No	Yes	Yes
File system access	No	Yes	Yes
Use external systems	No	Discouraged	Yes
Multiple threads	No	Yes	Yes
Sleep statements	No	Yes	Yes
System properties	No	Yes	Yes
Time limit (seconds)	60	300	900+

FIRST Principles - Fast

The screenshot shows the Xcode interface with a project named "BasicCalculator". The navigation bar indicates the build succeeded at 8:31 PM. The left sidebar shows the test suite structure:

- BasicCalculatorTests (3 tests)
 - BasicCalculatorTests (3 tests)
 - testAdd() (green checkmark)
 - testExample() (green checkmark)
 - testPerformanceExample() (green checkmark)

The main editor area displays the source code for the test suite:

```
override func setUp() {
    super.setUp()
    // Put setup code here. This method is called before the invocation of each test method in the class.
}

override func tearDown() {
    // Put teardown code here. This method is called after the invocation of each test method in the class.
    super.tearDown()
}

func testAdd() {
    XCTAssertEqual(resCalc.add(1, 1), 2)
    XCTAssertEqual(resCalc.add(1, 2), 3)
    XCTAssertEqual(resCalc.add(5, 4), 9)
}

func testExample() {
    // This is an example of a functional test case.
    XCTAssertTrue(true, "Pass")
}

func testPerformanceExample() {
    self.measureBlock() {
        XCTAssertEqual(self.resCalc.add(1, 2), 3)
    }
}
```

A red box highlights the `testPerformanceExample()` method. A performance analysis window is open on the right, titled "Performance Result". It shows the following details:

 - Metric: Time
 - Result: No Baseline
 - Average: 0.000s
 - Baseline: No Baseline
 - Max STDEV: 10.000%

A histogram shows the distribution of execution times. The log pane at the bottom shows the test session log and execution results:

```
Test session log:
/Users/admin/L
cauomeezodcnehenwy
Session-2016-04-18

Value: 0.00 (329.35%)
20:31:12.885.

Test Suite 'BasicC
Executed 1 test, with 0 failures (0 unexpected) in 0.283 (0.285) seconds
Test Suite 'Selected tests' passed at 2016-04-18 20:31:12.886.
Executed 1 test, with 0 failures (0 unexpected) in 0.283 (0.288) seconds
```

Performance Lab

1. Open Calculator project
2. Create new test file
3. Add assertion in Performance section
4. Test and measure speed.

FIRST Principles - Isolated

```
when(methodIsCalled).thenReturn(aValue);
```

```
// Cuckoo

stub(mock) { stub in
    when(stub.readWriteProperty.get).thenReturn(10)
}
```

Isolated - Cuckoo

The screenshot shows the Xcode interface with the project 'ETAMock' selected. The top status bar indicates 'Finished running ETAMock on iPhone 7 Plus'. The left sidebar lists project files: AppDelegate.swift, ViewController.swift, BusTableViewController.swift, RouteTableViewController.swift, StopsViewController.swift, JSONfetcher.swift, customJSONNparser.swift, Main.storyboard, Assets.xcassets, LaunchScreen.storyboard, Info.plist, Route.swift, ETAMockTests, GeneratedMocks.swift, ETAMockTests.swift, and Info.plist. The right pane displays the 'Build Phases' tab for the 'ETAMock' target. The 'Build Phases' section includes tabs for General, Resource Tags, Info, Build Settings, Build Phases (selected), and Build Rules. Under 'Build Phases', the 'TARGET Dependencies' section lists 'ETAMock'. The 'Build Phases' list contains the following items:

- > Target Dependencies (1 item)
- > [CP] Check Pods Manifest.lock
- > Compile Sources (2 items)
- > Link Binary With Libraries (1 item)
- > Copy Bundle Resources (0 items)
- > [CP] Embed Pods Frameworks
- > [CP] Copy Pods Resources
- > Run Script

The 'Run Script' section has a 'Shell' field set to '/bin/sh' and a script body:

```
1 # Define output file; change "$PROJECT_NAME}Tests" to your test's root source folder, if it's not the default name
```

Below the script, there are two checkboxes: 'Show environment variables in build log' (checked) and 'Run script only when installing' (unchecked). The bottom of the pane shows an 'Input Files' section.

Isolated - Cuckoo

```
# Define output file; change "${PROJECT_NAME}Tests" to your test's
root source folder, if it's not the default name

OUTPUT_FILE="./${PROJECT_NAME}Tests/GeneratedMocks.swift"
echo "Generated Mocks File = ${OUTPUT_FILE}"

# Define input directory; change "${PROJECT_NAME}" to your project's root
source folder, if it's not the default name
INPUT_DIR="./${PROJECT_NAME}"
echo "Mocks Input Directory = ${INPUT_DIR}"

# Generate mock files; include as many input files as you'd like to create mocks for
${PODS_ROOT}/Cuckoo/run generate --testable "${PROJECT_NAME}" \
--output "${OUTPUT_FILE}" \
"${INPUT_DIR}/FileName1.swift" \
"${INPUT_DIR}/FileName2.swift" \
"${INPUT_DIR}/FileName3.swift"
# ... and so forth
```

FIRST Principle - Repeatable

Post-build Actions

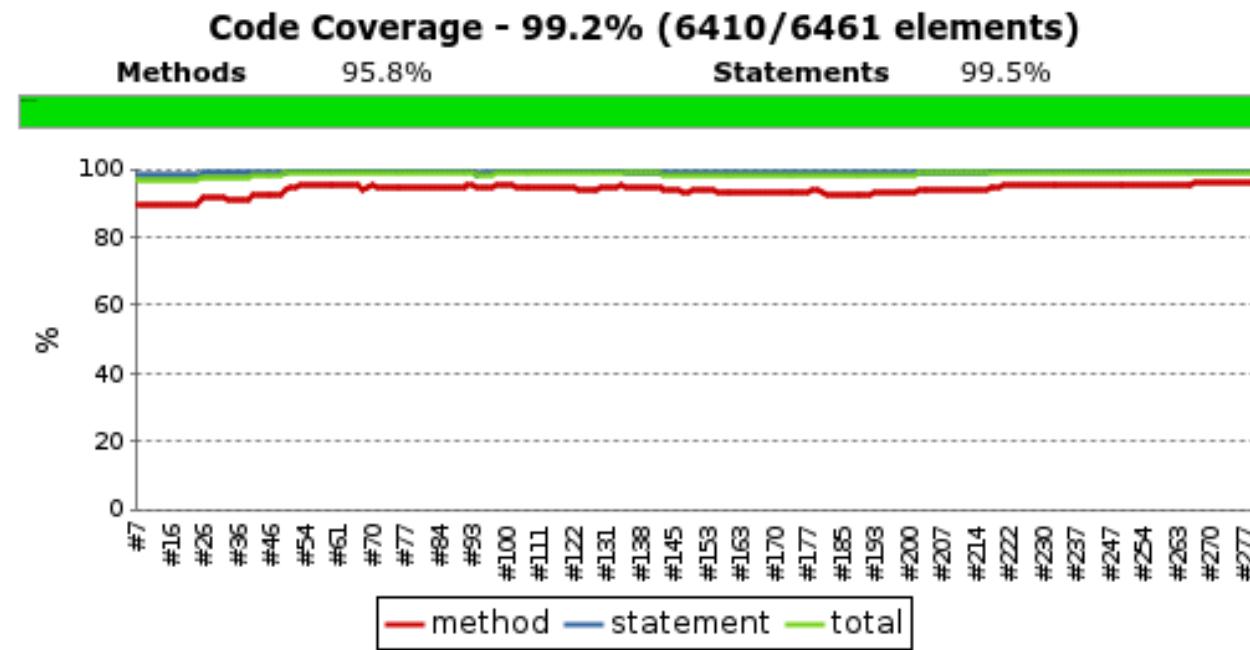
The screenshot shows a user interface for managing post-build actions. At the top, the title "Post-build Actions" is displayed. Below it, there is a list of actions, starting with "Delete workspace when build is done". To the right of this action is a "Advanced..." button. At the bottom of the list, there is a button labeled "Add post-build action ▾".

Delete workspace when build is done

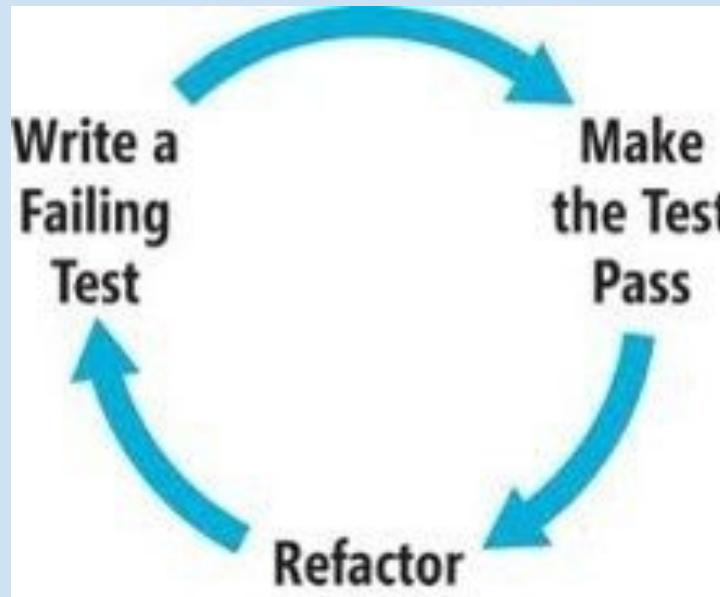
Advanced...

Add post-build action ▾

FIRST Prin - Self-Verifying



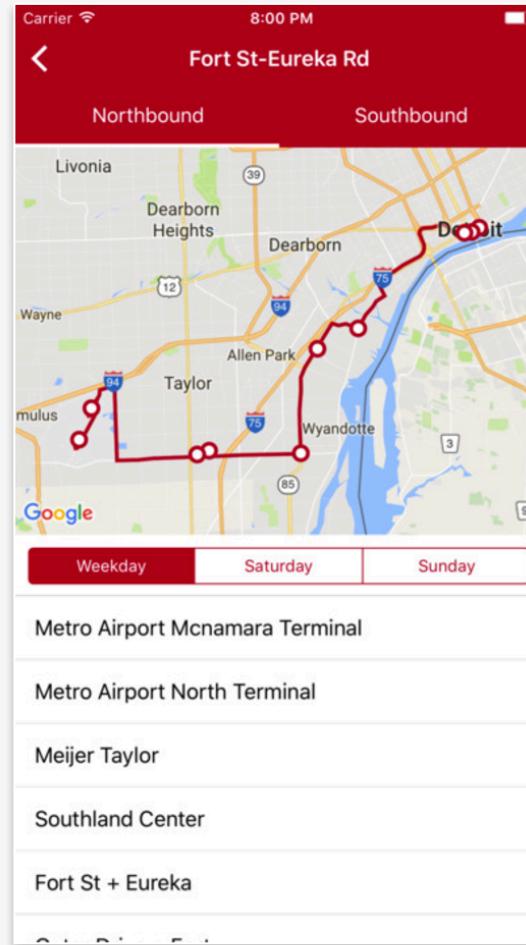
FIRST Principles - Timely



Sample App

Screenshots

iPhone | iPad



ROUTE	DETROIT ROUTE
7	Cadillac - Hamtramck
9	Chalmers
10	Chene
11	Clairmount
12	Conant
13	Conner
14	Crosstown
15	Chicago-Detroit
16	Dexter
17	Eight Mile
18	Fenkell
19	Fort
21	Grand River
22	Gratiot

Sample App

Carrier	5:55 PM	WiFi
Smart		
DDot		
Reflex		

Carrier	5:55 PM	WiFi
Back		
FORT ST-EUREKA RD		
SOUTHSHORE		
DOWNRIVER		
MICHIGAN AVENUE LOCAL		
FORD RD		
FORD RD EXPRESS		
TELEGRAPH		
MIDDLEBELT SOUTH		
GRAND RIVER-BEECH DALY		
SOUTHFIELD - ORCHARD RIDGE		
NORTHWESTERN HIGHWAY		
GREENFIELD-SOUTHFIELD		
GREENFIELD-SOUTHFIELD		
MAIN STREET-BIG BEAVER		

Carrier	5:56 PM	WiFi
Back		
Northb...	Southb...	
METRO AIRPORT MCNAMARA TERMINAL		
METRO AIRPORT NORTH TERMINAL		
MEIJER TAYLOR		
SOUTHLAND CENTER		
FORT ST + EUREKA		
OUTER DRIVE + FORT		
W JEFFERSON + COOLIDGE		
FORT + CASS		
LARNED + WOODWARD		

Sample App - Unit Test

```
func testParseRoutes() {
    let mock = MockJSONfetcher()

    stub(mock) { mock in
        when(mock.callApi(url: any(), completion: anyClosure())).then { url, closure in
            closure(self.testRouteJson)
        }
    }

    mock.callApi(url: url) { data in
        XCTAssertEqual(data, self.testRouteJson)
        let parser = customJSONparser(companyIndex: 1)
        let route = Route(name: "FORT ST-EUREKA RD",
                           direction1: "Northbound",
                           direction2: "Southbound", id: 1, routeId: "125")
        XCTAssertEqual(parser.getRoutes(fromJSONString: data), [route])
    }
}
```

```
xcodebuild test -workspace ETAMock.xcworkspace -scheme ETAMock
-destination 'platform=iOS Simulator, name=iPhone 7 Plus'
```

Sample App - Unit Test

```
# Define output file; change "${PROJECT_NAME}Tests" to your test's root source folder, if it's not the default name
OUTPUT_FILE="./${PROJECT_NAME}Tests/GeneratedMocks.swift"
echo "Generated Mocks File = ${OUTPUT_FILE}"

# Define input directory; change "${PROJECT_NAME}" to your project's root source folder, if it's not the default name
INPUT_DIR="./${PROJECT_NAME}"
echo "Mocks Input Directory = ${INPUT_DIR}"

# Generate mock files; include as many input files as you'd like to create mocks for
${PODS_ROOT}/Cuckoo/run generate --testable "${PROJECT_NAME}" \
--output "${OUTPUT_FILE}" \
"${INPUT_DIR}/JSONFetcher.swift"

# ... and so forth

# After running once, locate `GeneratedMocks.swift` and drag it into your Xcode test target group
```



```
  }
  ids.append(temp)
}
return ids
}

/// Parses a json string for a list of route object
///
/// - Parameter fromJSONString: the string to be parsed
/// - Returns: a list of route object
///
/// This makes it easier for us to use the data in the controllers
func getRoutes(fromJSONString: String) -> [Route] {

    // Convert string to data
    //guard let data = Data(fromJSONString, encoding: .utf8,
    //allowLossyConversion: false)
    //let json = try JSON(data: data).array else {
    //    return []
    //}

    do {

        let dataFromString = fromJSONString.data(using: .utf8,
            allowLossyConversion: false)
        let json = try JSON(data: dataFromString!).array

        // Go over each route and add it to the array
        var routes = [Route]()
        for routeJSON in json! {
            let name = routeJSON["routeName"].stringValue
            let direction1 = routeJSON["direction1"].stringValue
            let direction2 = routeJSON["direction2"].stringValue
            let id = routeJSON["id"].intValue
            let routeId = routeJSON["routeID"].stringValue
            let route = Route(name: name, direction1: direction1,
                direction2: direction2, id: id, routeId: routeId)
            routes.append(route)
        }
    }

    return routes
}
```

Isolated - Cuckoo Lab

0. sudo gem install cocoapods
1. git clone http://github.com/gnolanltu/ETAMock
2. View the podfile and see pod "Cuckoo" as a test target.
3. Run pod install.
4. Close the project and reopen the workspace.
5. Click on the project folder then choose Test Target ➤ Build Phases.
6. Click + and choose New Run Script Phase.
7. Add Listing to the Run Script section, making sure to modify the input files that you want to mock.
8. Build the project.
9. Run the tests.
10. Drag and drop GeneratedMocks.swift into the test section.
11. Uncomment out your mocked tests
12. Run the mocked tests.

Sample App - API Test

```
← → ✎ ⌂ ec2-204-236-211-33.compute-1.amazonaws.com:8080/companies/3/routes
[{"company": {"id": 3, "name": "Reflex", "brandcolor": "#498BC5", "busImgURL": "http://ec2-204-236-211-33.compute-1.amazonaws.com:8080/assets/images/Reflex-Bus.png", "logoImgURL": null}, "companyID": 3, "routeID": "DDOT_6455", "routeName": "REFLEX", "routeNumber": "498", "direction1": "northbound", "direction2": "southbound", "daysActive": "Weekday,Saturday,Sunday", "id": 86}, {"company": {"id": 3, "name": "Reflex", "brandcolor": "#498BC5", "busImgURL": "http://ec2-204-236-211-33.compute-1.amazonaws.com:8080/assets/images/Reflex-Bus.png", "logoImgURL": null}, "companyID": 3, "routeID": "598", "routeName": "GRATIOT REFLEX to DETROIT", "routeNumber": "598", "direction1": "NORTHBOUND", "direction2": "SOUTHBOUND", "daysActive": "Weekday,Saturday,Sunday", "id": 87}, {"company": {"id": 3, "name": "Reflex", "brandcolor": "#498BC5", "busImgURL": "http://ec2-204-236-211-33.compute-1.amazonaws.com:8080/assets/images/Reflex-Bus.png", "logoImgURL": null}, "companyID": 3, "routeID": "599", "routeName": "GRATIOT REFLEX to DMC/WSU", "routeNumber": "599", "direction1": "NORTHBOUND", "direction2": "SOUTHBOUND", "daysActive": "Weekday,Saturday,Sunday", "id": 88}]
```

Sample App - API Test

The screenshot shows the Postman application interface. At the top, there's a header with 'GET' selected, a URL input field containing 'http://ec2-204-236-211-33.compute-1.amazonaws.com:8080/companies/1/routes/1/Northbound/weekday/1/stops', and buttons for 'Params', 'Send', 'Save', 'Cookies', and 'Code'. Below the header, the 'Authorization' section is visible. The main area contains a code editor with the following JSON test script:

```
1 tests["Body has stopID"] = responseBody.has("stopID");
2 tests["Body has stopName"] = responseBody.has("stopName");
3 tests["Body has latitude"] = responseBody.has("latitude");
4 tests["Body has longitude"] = responseBody.has("longitude");
5
6 var jsonData = JSON.parse(responseBody);
7
8 tests["The first stopID equals String 22362"] = jsonData[0]
   .stopID === "22362";
9 tests["The first stopName equals String METRO AIRPORT
   MCNAMARA TERMINAL"] = jsonData[0].stopName === "METRO
   AIRPORT MCNAMARA TERMINAL";
10 tests["The first order equals String 1"] = jsonData[0].order
    === 1;
```

To the right of the code editor is a sidebar titled 'SNIPPETS' containing several items:

- Clear a global variable
- Clear an environment variable
- Response body: Contains string
- Response body: Convert XML body to a JSON Object
- Response body: Is equal to a string
- Response body: JSON value check
- Response headers: Content-Type header check
- Response time is less than 200ms
- Set a global variable

```
newman run ETAJson.postman_collection
```

Sample App - API Test

RIIS-MBPRO-13:Android User\$ newman run ETAJson.postman_collection.json newman																				
ETAJson																				
→ http://ec2-204-236-211-33.compute-1.amazonaws.com:8080/companies/1/routes/1/Northbound/weekday/1/stops GET http://ec2-204-236-211-33.compute-1.amazonaws.com:8080/companies/1/routes/1/Northbound/weekday/1/stops [200 OK, 1.74KB, 103ms] ✓ Body has stopID ✓ Body has stopName ✓ Body has latitude ✓ Body has longitude ✓ The first stopID equals String 22362 ✓ The first stopName equals String METRO AIRPORT MCNAMARA TERMINAL ✓ The first order equals String 1																				
<table border="1"><thead><tr><th></th><th>executed</th><th>failed</th></tr></thead><tbody><tr><td>iterations</td><td>1</td><td>0</td></tr><tr><td>requests</td><td>1</td><td>0</td></tr><tr><td>test-scripts</td><td>1</td><td>0</td></tr><tr><td>prerequest-scripts</td><td>0</td><td>0</td></tr><tr><td>assertions</td><td>7</td><td>0</td></tr></tbody></table>				executed	failed	iterations	1	0	requests	1	0	test-scripts	1	0	prerequest-scripts	0	0	assertions	7	0
	executed	failed																		
iterations	1	0																		
requests	1	0																		
test-scripts	1	0																		
prerequest-scripts	0	0																		
assertions	7	0																		
total run duration: 152ms																				
total data received: 1.59KB (approx)																				
average response time: 103ms																				

Sample App - GUI Test

```
func testExample() {
    let app = XCUIApplication()
    let tablesQuery = app.tables
    tablesQuery.staticTexts["Smart"].tap()
    tablesQuery.staticTexts["SOUTHSHORE"].tap()
    app.buttons["Southbound"].tap()
    XCTAssert(tablesQuery.staticTexts["JEFFERSON + SOUTHFIELD"].exists)

    app.navigationBars["ETAMock.StopsView"].children(matching: .button)
        .matching(identifier: "Back").element(boundBy: 0).tap()
    tablesQuery.staticTexts["MICHIGAN AVENUE LOCAL"].tap()
    app.buttons["Westbound"].tap()
    XCTAssert(tablesQuery.staticTexts["MICHIGAN + CASS"].exists)
}
```

```
xcodebuild build -workspace ETAMock.xcworkspace -scheme ETAMock
-destination 'platform=iOS Simulator, name=iPhone 7 Plus'
```

Sample App - Jenkins

Build

Execute shell

Command `#!/bin/bash -l
pod install`

X

?

See [the list of available environment variables](#)

Advanced...

Execute shell

Command `export PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/local/bin
/usr/local/Cellar/node/7.8.0/libexec/npm/bin/newman run ETATJson.postman_collection.json`

X

?

See [the list of available environment variables](#)

Advanced...

Execute shell

Command `xcodebuild build -workspace ETAMock.xcworkspace -scheme ETAMock -destination 'platform=iOS Simulator,name=iPhone 7 Plus'`

X

?

See [the list of available environment variables](#)

Sample App - Jenkins

← → ⌂ ⓘ 127.0.0.1:8080/job/ETAMockLTU/ ⋮

 Jenkins ⚡ 2 ⚡ search ⚡ Godfrey ⚡ log out ⚡

Jenkins > ETAMockLTU > ⚡ ENABLE AUTO REFRESH

Back to Dashboard Status Changes Workspace Build Now Delete Project Configure GitHub

Project ETAMockLTU

 [Workspace](#)

 [Recent Changes](#)

[add description](#) [Disable Project](#)

Permalinks

- [Last build \(#10\), 15 days ago](#)
- [Last stable build \(#10\), 15 days ago](#)
- [Last successful build \(#10\), 15 days ago](#)
- [Last failed build \(#6\), 20 days ago](#)
- [Last unsuccessful build \(#7\), 20 days ago](#)
- [Last completed build \(#10\), 15 days ago](#)

Build History trend ⚡

find
#10 Apr 18, 2017 6:00 PM
#9 Apr 18, 2017 1:45 PM
#8 Apr 15, 2017 8:25 AM

Sample App - Jenkins

← → ⌂ 127.0.0.1:8080/job/ETAMockLTU/10/consoleFull

 Jenkins 2 search Godfrey | log out

Jenkins > ETAMockLTU > #10

 Back to Project
 Status
 Changes
 Console Output
 View as plain text
 Edit Build Information
 Delete Build
 Git Build Data
 No Tags
 Previous Build

Console Output

```
Started by user Godfrey
Building in workspace /Users/Shared/Jenkins/Home/workspace/ETAMockLTU
Cloning the remote Git repository
  Cloning repository https://github.com/gnolanltu/ETAMock
    > git init /Users/Shared/Jenkins/Home/workspace/ETAMockLTU # timeout=10
Fetching upstream changes from https://github.com/gnolanltu/ETAMock
  > git --version # timeout=10
  > git fetch --tags --progress https://github.com/gnolanltu/ETAMock +refs/heads/*\:refs/remotes/origin/*
  > git config remote.origin.url https://github.com/gnolanltu/ETAMock # timeout=10
  > git config --add remote.origin.fetch +refs/heads/*\:refs/remotes/origin/* # timeout=10
  > git config remote.origin.url https://github.com/gnolanltu/ETAMock # timeout=10
Fetching upstream changes from https://github.com/gnolanltu/ETAMock
  > git fetch --tags --progress https://github.com/gnolanltu/ETAMock +refs/heads/*\:refs/remotes/origin/*
  > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
  > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision bd9475793dc29169da6164b4addirf4915f1f5a942 (refs/remotes/origin/master)
  > git config core.sparsecheckout # timeout=10
  > git checkout -f bd9475793dc29169da6164b4addirf4915f1f5a942
  > git rev-list bd9475793dc29169da6164b4addirf4915f1f5a942 # timeout=10
[ETAMockLTU] $ /bin/sh -xe /Users/Shared/Jenkins/tmp/hudson2310462430373189176.sh
+ export PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/local/bin
+ PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/local/bin
+ /usr/local/bin/newman run /Users/Shared/Jenkins/Home/workspace/ETAMockLTU/ETAJson.postman_collection.json
newman

ETAJson

→ http://ec2-204-236-211-33.compute-1.amazonaws.com:8080/companies/1/routes/1/Northbound/weekday/1/stops
GET http://ec2-204-236-211-33.compute-1.amazonaws.com:8080/companies/1/routes/1/Northbound/weekday/1/stops [200 OK, 1.71KB, 113ms]
✓ Body has stopID
✓ Body has stopName
✓ Body has latitude
✓ Body has longitude
✓ The first stopID equals String 22362
✓ The first stopName equals String METRO AIRPORT MCNAMARA TERMINAL
✓ The first order equals String 1
```

SonarQube - Jenkins

Dashboards Projects ▾ Measures Issues Rules Quality Profiles Quality Gates Log in Search

Swift :: Calculator Project >

Dashboard

Issues Time Machine

TOOLS Components Issues Drilldown Design Libraries Compare



Version 1.0 - May 22 2016 15:31

Lines Of Code 91	Files 3	Functions 16	SQALE Rating A	Technical Debt Ratio 1.1%	
Swift	Directories 1	Lines 156	Classes 3	Statements 66	
Duplications 0.0%	Lines 0	Blocks 0	Files 0		

Complexity
1.4 /function
7.7 /class
7.7 /file
Total: **23**

Events All May 22 2016 Version 1.0

Project calculator

-  SonarQube
-  Workspace
-  Recent Changes

SonarQube Quality Gate

SonarQube - Jenkins

The screenshot shows the SonarQube interface for a Swift project named "Calculator Project". The top navigation bar displays the project name, file path "Calculator/ViewController.swift", and key metrics: 48 Lines of code, A Debt, 5min Issues, and 5 Issues.

Time Changes section:

- Filters: Unresolved Issues, Open/Reopened Issues, Fixed Issues, False Positive Issues.
- Severities: Major (selected), Minor, Critical.
- Rules: Useless parentheses around ...

Code Review section:

```
21
22     override func viewDidLoad() {
23         super.viewDidLoad()
24         operand = "="
25         resultsFld.text = ("\"\\(res)\"")
```

A tooltip for line 25 suggests: **Remove those useless parentheses**. It also shows a link to "Open" and a note about "Debt: 1min".

```
26
27
28     override func didReceiveMemoryWarning() {
29         super.didReceiveMemoryWarning()
```

```
31     }
32
33     @IBAction func compute(sender: UIButton) {
34         num = num * 10 + Int(sender.titleLabel!.text!)!
35         resultsFld.text = ("\"\\(num)\"")
```

SonarQube - Jenkins

SonarQube servers

Environment variables

SonarQube installations

Enable injection of SonarQube server configuration as build environment variables
If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Name: SwiftSonar

Server URL: http://localhost:9000

Default is http://localhost:9000

Server version: 5.3 or higher

Configuration fields depend on the SonarQube server version.
Server authentication token: |

SonarQube authentication token. Mandatory when anonymous access is disabled.

SonarQube account login: [disabled]

SonarQube account password: [disabled]

SonarQube account used to perform analysis. Mandatory when anonymous access is disabled. No longer used since SonarQube 5.3.

SonarQube account used to perform analysis. Mandatory when anonymous access is disabled. No longer used since SonarQube 5.3.

[Advanced...](#)

Code signing & OS X keychain options

- [Execute SonarQube Scanner](#)
- [Execute Windows batch command](#)
- [Execute shell](#)
- [Import developer profile](#)
- [Invoke Ant](#)
- [Invoke top-level Maven targets](#)
- [SonarQube Scanner for MSBuild - Begin Analysis](#)
- [SonarQube Scanner for MSBuild - End Analysis](#)
- [Xcode](#)

[Add build step](#) ▾

SonarQube Scanner

SonarQube Scanner installations

SonarQube Scanner

Name

SwiftSonar

SONAR_RUNNER_HOME

/etc/sonar-runner/

Install automatically

[Delete SonarQube Scanner](#)

[Add SonarQube Scanner](#)

List of SonarQube Scanner installations on this system



Fastlane

1. brew cask install fastlane
2. Run in SimpleETA project directory
 fastlane init
3. Answer questions
4. Run
 fastlane test

URLs

<http://riis.com/blog>

<https://getpostman.com>

<https://github.com/postmanlabs/newman>

<https://github.com/Brightify/Cuckoo>

<https://jenkins.io>

<http://www.cimgf.com/2015/05/26/setting-up-jenkins-ci-on-a-mac-2/>

<https://github.com/gnolanltu/ETAMock>

<https://sonarqube.org>

<https://github.com/fastlane/fastlane>

CONTACT INFO

godfrey@riis.com
@godfreynolan

